SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

TYÖAIKASEURANTA-SOVELLUKSEN UUDEN KÄYTTÖLIITTYMÄN SUUNNITTELU JA TOTEUTUS

TEKIJÄ: Karoliina Jäppinen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala			
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma			
Työn tekijä(t) Karoliina Jäppinen			
Työn nimi Työaikaseuranta-sovelluksen uuden käyttöliittymän suunnittelu ja toteutus			
Päiväys	17.05.2020	Sivumäärä/Liitteet	25
Toimeksiantaja/Yhteistyökumppani(t) Documtec Oy			

Tiivistelmä

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa uusi käyttöliittymä työajanseuranta -sovellukselle. Työn tilaajana toimi Documtec Oy. Documtec on tietotekniikka-alan yritys Pieksämäellä. VISION Taika on työajan seuraamiseen tarkoitettu ohjelmisto, joka on myös yksi Documtecin tarjoamista tuotteista. Työlle oli todellinen tarve, sillä Taikan nykyinen versio ei tue mobiililaitteita.

Työn suunnitteluvaiheessa perehdyttiin työssä käytettäviin tekniikoihin ja Taikan vanhan version koodiin sekä sen toimintaan. Lisäksi työn aihe rajattiin käyttöliittymän pohjan sekä seurantatoiminnon kehittämiseen. Suunnittelussa otettiin myös huomioon työn tilaajan toive selkeästä ja tulevaisuudessa helposti ylläpidettävästä koodista.

Käyttöliittymän kehitys jaettiin työtehtäviin, jotka toteutettiin kehityksen kannalta järkevissä kokonaisuuksissa. Käyttöliittymälle rakennettiin sivujen välinen navigointi, teema sekä sisäänkirjautumis- ja seurantatoiminnot. Toiminnan testauksessa käytettiin apuna testaukseen tarkoitettua dataa.

Uusi käyttöliittymä tehtiin Xamarin.Forms-tekniikan avulla, koska se mahdollistaa Android-, iOs- ja Windows-sovellusten kehittämisen yhdellä yhteisellä koodipohjalla. Xamarin.Forms käyttää ohjelmointikielinä C#:a sekä käyttöliittymän ulkoasun rakentamisessa XAML:a.

Opinnäytetyön lopputuloksena saatiin uudistettu käyttöliittymän pohja Taika-ohjelmistolle, joka toimii myös Android-pohjaisilla mobiililaitteilla. Käyttöliittymään lisättiin myös työajanseurantatoiminto, jolla työntekijä pystyy kirjautumaan sisään ja ulos töistä. Lisäksi työtehtävän pystyy valikoimaan sisään kirjautuessa tai vaihtamaan tarvittaessa myös kesken päivän. Uusi käyttöliittymä tulee työn tilaajalle lopulta käyttöön korvaamalla vanhan.

Avainsanat
Työajanseuranta, käyttöliittymä, Xamarin.Forms

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Karoliina Jäppinen			
Title of Thesis Design and development of a new user interface for working time monitoring software			
Date	17 May 2020	Pages/Appendices	25
Client Organisation /Partners Documtec Oy			

Abstract

The purpose of the thesis was to design and develop a new user interface for the working time monitoring application. The work was commissioned by Documtec Oy. Documtec is an information technology company in Pieksämäki. VISION Taika is software for monitoring working hours, which is also one of the products offered by Documtec. There was a real need for the work, because the current version of Taika does not support mobile devices.

During the design phase of the thesis work, the techniques used in the thesis work and the code and its operations of the old version of Taika, were studied. In addition, the topic of the thesis work was limited to the development of the base of the user interface and the monitoring function. The customer's wish for a clear code and for a code that will be easy to maintain in the future was also taken into account. The development of the user interface was divided into work tasks, which were carried out in development sensible entities. Inter page navigation, a theme and login and tracking functions were built for the user interface. Data what was made for testing were used to test the operation of the user interface. The new user interface was made by using Xamarin.Forms technology because it allows the development of Android, iOs and Windows applications which all have the same code base. Xamarin.Forms uses C# as its programming language and XAML to build the layout of the user interface.

The result of the thesis was a renewed user interface base for Taika software, which also works on Android based mobile devices. A working time monitoring function was also added to the user interface, which allows the employee to log in and out of work. In addition, the work task can be selected at login or changed, if necessary, during the day. The commisioner of the thesis will eventually deploy the new interface by replacing the old one.

Keywords
Working time monitoring, user interface, Xamarin.Forms

ESIPUHE

Haluaisin kiittää työni tilaajaa Documtecia siitä, että sain toteuttaa opinnäytetyöni heille. Erityisesti haluan kiittää yrityksessä toimivaa asiantuntijaa, Pauli Huuskosta, joka jaksoi antaa neuvoja sekä apua aina tarvittaessa.

Kiitos myöskin työni ohjaajalle Jukka Kinnuselle. Häneltä sain palautetta sekä ohjeistusta työn tekemisen aikana.

Lopuksi haluan vielä kiittää läheisiäni, joilta sain tukea ja kannustusta opinnäytetyöhöni.

Kuopiossa 27.04.2020
Karoliina Jäppinen

TERMIT JA LYHENTEET

.NET
Microsoftin kehittämä avoimen lähdekoodin kehittäjäalusta, jolla pystyy kehittämään erilaisia sovelluksia useisiin eri ympäristöihin. .NET tukee myös monia kirjastoja sekä ohjelmointikieliä, C#, F# ja Visual Basic. (.NET)

UI
User Interface eli käyttöliittymä on sovelluksen osa, jota sovelluksen käyttäjä käyttää. Käyttöliittymä on usein visuaalinen kokonaisuus, jossa on esimerkiksi kuvia, nappeja ja erilaisia tekstikenttiä. Käyttöliittymän kautta tapahtuu käyttäjän ja sovelluksen välinen vuorovaikutus.

Rajapinta
Rajapinnan avulla sovellukset voivat siirtää tietoa keskenään. Tieto siirtyy kevyessä muodossa esimerkiksi JSON-tyyppisenä rajapinnan läpi, jonka jälkeen sovelluksen osat muuntavat tiedon tarvitsemaansa muotoon ja jatkokäsittelevät sitä (Katso kohta 3.8.2 Newtonsoft.JSON).

Client
Tarkoittaa asiakasohjelmaa, tässä tapauksessa asiakasohjelmana toimii Taikan käyttöliittymä. Asiakasohjelma lähettää ja vastaanottaa tietoa ohjelman palvelinpuolen kanssa rajapintojen avulla.

Android, iOs, Windows
Ovat käyttöjärjestelmiä. Android ja iOs ovat moobiililaitteille tehtyjä käyttöjärjestelmiä. Android on Linux-pohjainen käyttöjärjestelmä, jota suositut mobiililaitteiden tekijät, Samsung ja Huawei, käyttävät laitteissaan. iOs puolestaa on Applen kehittämä käyttöjärjestelmä heidän puhelimiaan varten. Windows on Microsoftin kehittämä ja sitä käytetään esimerkiksi tietokoneissa, puhelimissa ja Xbox-pelikonsoleissa.

SISÄLTÖ

1 JOHDANTO

Documtec Oy on vuonna 2006 perustettu IT-alan yritys. Documtec sijaitsee Pieksämäellä ja siellä työskentelee kymmenkunta henkilöä. Documtecin palvelut voidaan jakaa kahteen kokonaisuuteen, WITH- ja VISION-palveluihin. WITH-palvelu tarjoaa muille yrityksille IT-ympäristön rakentamista ja sen ylläpitoa. VISION-palvelu puolestaan tarjoaa erilaisia yritystoimintaa parantavia tuotteita.

VISION Taika on yksi Documtecin tarjoamista tuotteista. Taika on työajanseurantaan tarkoitettu ohjelmisto. Taika on käytössä monilla Documtecin asiakkailla, sekä myös heillä itsellään. Ohjelmisto toimii pääasiallisesti Windows-pohjaisilla kosketusnäytön omaavilla laitteilla. (Documtec)

Opinnäytetyön aihe syntyi yrityksen tarpeesta saada Taika toimimaan myös Android-pohjaisilla mobiililaitteilla. Työn tarkoituksena ja tavoitteena oli suunnitella ja kehittää kyseiselle ohjelmistolle uusi käyttöliittymä. Uutta käyttöliittymää tulee pystyä käyttämään kaiken kokoisilla näytöillä sekä uutena ominaisuutena myös Android-pohjaisilla mobiililaitteilla.

Opinnäytetyössä esitellään Taika-ohjelmiston toimintaa, uuden käyttöliittymän kehityksessä käytettävät tekniikat ja toteutuksen vaiheet. Lopussa on myös yhteenveto sekä hieman pohdintaa työn onnistumisesta.

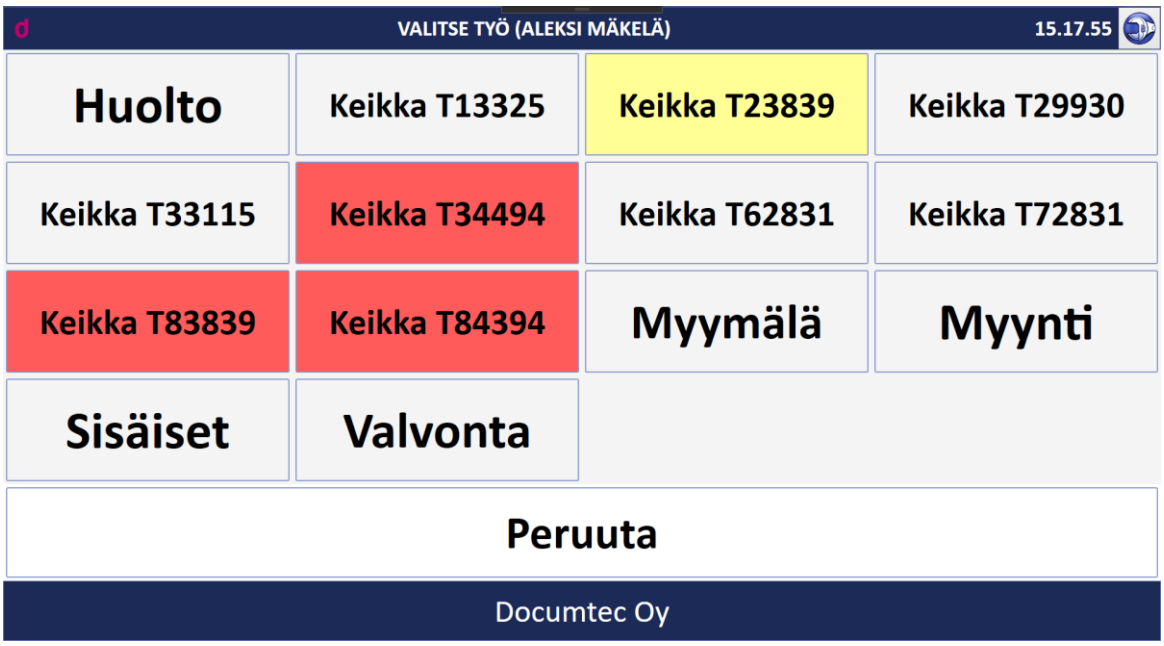
2 TAIKA-OHJELMISTO

VISION Taika on työajanseurantaan tarkoitettu ohjelmisto. Siitä on olemassa kaksi versiota, yritys ja yksilöllinen. Versio määräytyy käytettävän lisenssin avulla. Taikan yritysversio on tarkoitettu käytettäväksi työpaikalla yleisessä käytössä olevalta laitteelta. Yksillöllinen versio on nimensä mukaan tarkoitettu yhden henkilön käytettäväksi. Se asennetaan henkilön omaan laitteeseen ja sitä on tarkoitus käyttää työajan seuraamiseen esimerkiksi etätöitä tehdessä. Seuraavissa kappaleissa käydään tarkemmin läpi Taikan toimintoja sekä rakennetta.

KUVA 1. Taikan yritysversion perusnäkymä, nimet ovat testausta varten keksittyjä.

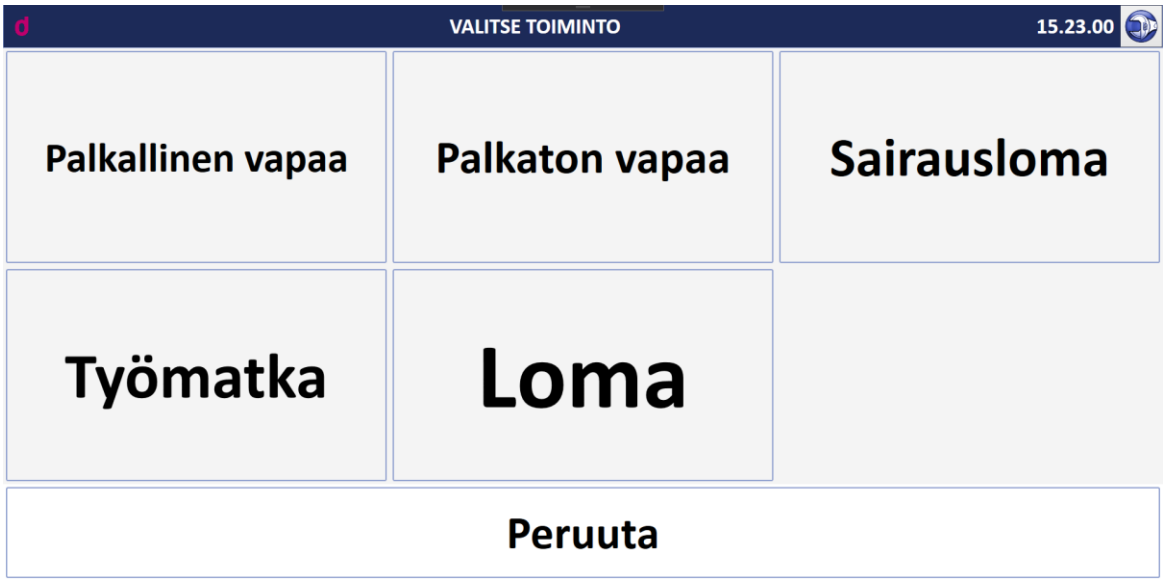
2.1 Toiminnot

Taika-ohjelmiston perustoiminto on työajan seuraaminen. Sisään tai ulos kirjautuessa työntekijä valitsee listasta itsensä. Käyttöliittymän perusnäkymä on tehty yksinkertaiseksi ja siinä näkyy valittavien työntekijöiden nimet, kellonaika, logo sekä yrityksen nimi (Kuva 1). Valinnan voi myös suorittaa kasvojentunnistusta käyttämällä, kuvassa vasen ylänurkka. Sisään kirjautuessa työntekijä valitsee myös tehtävän työtehtävien listasta (Kuva 2). Kesken päivän on myös mahdollista vaihtaa työtehtävästä toiseen. Työtehtävät pystytään määrittämään jokaisen asiakkaan tarkoituksiin sopiviksi.



KUVA 2. Työtehtävien valinta sisäänkirjautumisen yhteydessä

Taika sisältää useita toimintoja työajanseurannan lisäksi, kuten lomien ja työmatkan kirjaaminen (Kuva 3). Lisäksi Taikasta löytyy myös omien tietojen katselu- ja muokkaustoiminnot. Omien tietojen katselutoiminnon avulla henkilö pystyy tarkastelemaan oman työaikansa liukumia. Muokkauksessa puolestaan henkilö pystyy vaihtamaan oman Taikassa käytettävän Pin-koodinsa uuteen. Omien tietojen osuudessa on myös mahdollista lähettää viestejä. Kaikki viestit menevät yhteen määrättyyn sähköpostiosoitteeseen. Viestin lähetystoiminto on kehitetty sitä varten, että sitä kautta voi ilmoittaa Taikassa olevista ongelmista vioista vastaavalle henkilölle. Tälläinen ongelma voi olla esimerkiksi työtehtävän puuttuminen.



KUVA 3. Muiden tietojen valikko

2.2 Rakenne

Taikan nykyinen käyttöliittymä on rakennettu WPF-mallia hyödytäen. WPF eli Windows Presentation Foundation on Microsoftin kehittämä käyttöliittymäkehys, minkä avulla pystytään kehittämään työpöytäsovelluksia. WPF käyttää ohjelmointikielenä C#:a (Katso kohta 3.1 C#) sekä XAML-merkintäkieltä (Katso kohta 3.2 XAML) käyttöliittymän ulkoasun rakentamiseen. (WPF)

Käyttöliittymä toimii clienttina eli asiakasohjelmana, joka vaihtaa tietoja ohjelmiston palvelinpuolen kanssa. Palvelimen puolelle on rakennettu rajapinnat tiedon siirtämistä varten. Client-ohjelmasta kutsutaan näitä rajapintoja käyttäjän tekemien valintojen mukaisesti.

Taikan toimintaperiaate on tyhjennä ja lataa. Aina näkymän vaihtuessa edellinen tyhjenee ja uusi ladataan tilalle. Käyttäjän tekemä valinta määrittää, mitä seuraavaan näkymään tulee näkyviin. Kaikki käyttöliittymän sisältö tulee palvelimelta. Käyttöliittymässä on määritelty vain sen osat ja palvelin muodostaa näistä osista kokonaisen näkymän.

Käyttöliittymä koostuu kolmesta osiosta: ylä- ja alapaneelit, isompi ylänäkymä sekä pienempi alanäkymä. Yläpaneelissa on yrityksen logo, tekstikenttä, kello ja lisätoiminnot-nappi. Alapaneelissa puolestaan on vain tekstikenttä. Isompi ylänäkymä pitää sisällään valittavat tiedot, esimerkiksi työntekijöiden nimet tai työtehtävät. Pienemmässä alanäkymässä on valittavana toiminto, kuten peruuta tai valmis. (Kuvat 1 ja 2)

Taikaan on lisätty myös kasvojentunnistustoiminto. Se on rakennettu Microsoft Azuren tekoälyä hyödyntäen. Kasvojentunnistus on osa Azuren Cognitive -palveluita. Cognitive-palvelut tarjoavat tekoälyratkaisuja myös moniin muihin käyttötarkoituksiin, kuten puheen- ja kielenymmärtämiseen. (CognitiveServices)

3 KÄYTETYT TEKNIIKAT

Opinnäytetyössä käytettävät tekniikat määräytyivät Documtecin käytäntöjen mukaan. Visual Studio toimii kehitysympäristönä kaikissa kehitystöissä, joihin se soveltuu. Azure DevOps -versionhallinta on myös käytössä jokaisessa projektissa. Xamarin.Forms ja UWP -tekniikat valittiin juuri tähän tarkoitukseen sopivimmiksi. Niiden avulla on mahdollista rakentaa samaa koodipohjaa käyttävä sovellus, joka toimii sekä Windows- että Android-pohjaisilla laitteilla. Käytettävät ohjelmointikielet valikoituivat Xamarin.Forms-tekniikan mukaan ja ne ovat C# sekä XAML.

3.1 C# -kieli

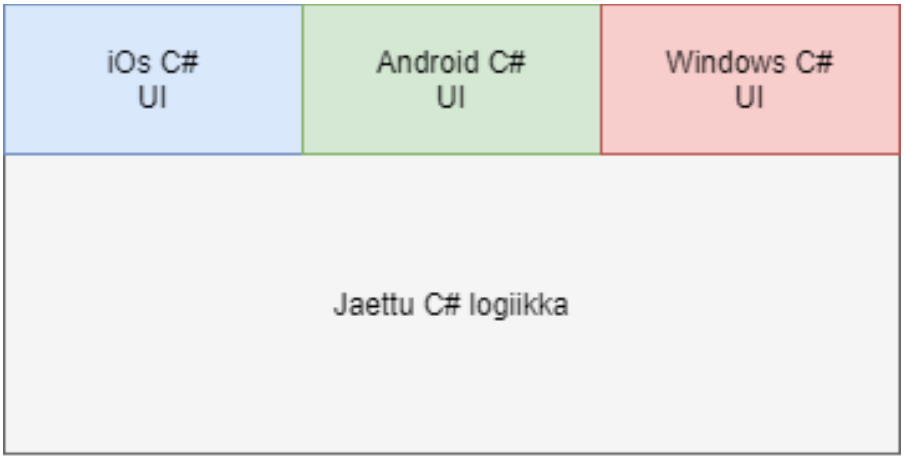
C# on Microsoftin kehittämä ohjelmointikieli. C# on helppokäyttöinen olio-ohjelmointikieli, jonka rakentamiseen on käytetty pohjana useita muita kieliä, kuten C++:aa ja Javaa. Lisäksi C# on järjestelmäriippumaton, eli sen avulla pystyy kehittämään ohjelmia monille ympäristöille, esimerkiksi Windows-, Linux- tai pilviympäristöille. (C#)

3.2 XAML-kieli

XAML on Microsoftin kehittämä merkintäkieli. XAML:n tarkoitus on helpottaa käyttöliittymien rakentamista. Käyttöliittymän yhdistäminen kooditaustatiedostoihin on myös mahdollista XAML:n avulla, tämä helpottaa käyttöliittymän logiikan rakentamista. (XAML)

3.3 Xamarin

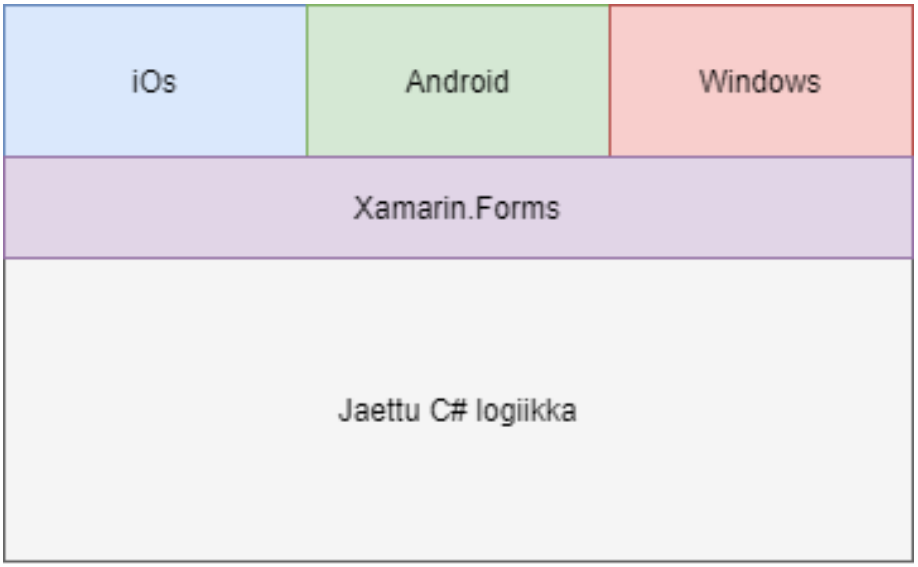
Xamarin on Microsoftin kehittämä sovellusalusta. Sitä hyödyntäen voidaan rakentaa natiiveja sovelluksia esimerkiksi Androidille ja iOs:lle käyttäen .Net:a ja C#:a. Xamarinissa on yhteinen jaettu koodipohja, joka on kirjoitettu C#:lla. Käyttöliittymät puolestaan toteutetaan kullekin ympäristölle erikseen alustakohtaisesti (Kaavio 1). Natiivisovellus tarkoittaa sitä, että se on kehitetty jotakin tiettyä ympäristöä tai laitetta varten. (Xamarin)



KAAVIO 1. Xamarinin toiminnan havainnollistaminen (Xamarin)

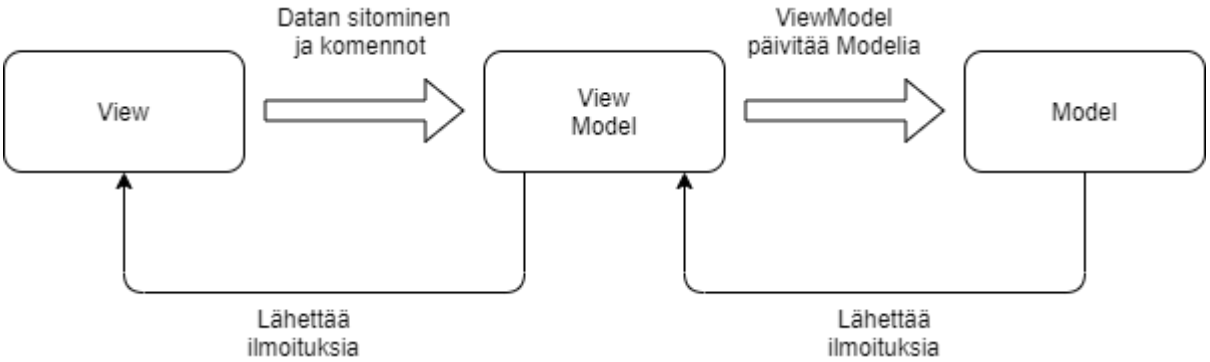
3.4 Xamarin.Forms

Xamarin.Forms on käyttöliittymäkehys, jonka avulla voidaan rakentaa natiiveja Android-, iOs- ja Windows-sovelluksia. Xamarin.Forms käyttää ohjelmointikielenä C#:a sekä käyttöliittymässä XAML-merkintäkieltä. Xamarinista poiketen Xamarin.Formsin avulla eri alustat jakavat myös yhteisen koodipohjan käyttöliittymää varten (Kaavio 2). (Xamarin.Forms)



KAAVIO 2. Xamarin.Forms toiminnan havainnollistaminen (Xamarin)

Xamarin.Forms tukee yleistä MVVM (Model-View-ViewModel) -ohjelma-arkkitehtuuria. MVVM:n avulla on helppoa pitää sovelluksen logiikka ja käyttöliittymä erillään. Tämä tekee koodista myös helposti uudelleenkäytettävää, muokattavaa sekä ylläpidettävää. (MVVM)



KAAVIO 3. MVVM:n toiminta (MVVM)

3.4.1 View (Näkymä)

View:n eli näkymän tehtävänä on vastata käyttöliittymän rakenteesta sekä ulkoasusta. Näkymällä voi olla myös käytöliittymää ohjaavaa logiikkaa taustalla. Ulkoasu määritellään XAML-kielellä, kun taas taustakooditiedostot kirjoitetaan C#-kielellä. XAML-tiedostossa määritellään ulkoasun komponenttejen ominaisuudet, kuten koko, väri ja paikka käyttöliittymässä (Kuvat 4 ja 5). Näkymä keskustelee näkymämallin kanssa käyttäen hyväksi datan sitomista ja komentoja (Kaavio 3). (MVVM)

```
<ContentView xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="TAikaSeuranta.XClient.Views.StaticInfoView"
             xmlns:models="clr-namespace:TAikaSeuranta.XClient.Models"
             models:ViewModelLocator.AutoWireViewModel="True">
   <ContentView.Content>
          <Frame CornerRadius="2"
                 BorderColor="{StaticResource basicColor2}"
                 BackgroundColor="{StaticResource basicColor1}"
                 HorizontalOptions="CenterAndExpand"
                 VerticalOptions="CenterAndExpand">
          <Grid HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand">
                 <Label
                   Text="{Binding Text, FallbackValue=CONTENT}"
                   Margin="8"
                   TextColor="{StaticResource basicTextColor1}"
                   HorizontalTextAlignment="Center"
                   VerticalTextAlignment="Center">
                 </Label>
             </Grid>
          </Frame>
   </ContentView.Content>
</ContentView>
```

KUVA 4. Esimerkki yksinkertaisesta XAML-kielellä kirjoitetusta näkymästä

KUVA 5. Kuvan 4 XAML-koodin tuotos

3.4.2 View Model (Näkymämalli)

View Modelin eli näkymämallin tehtävänä on yhdistää Modelin eli mallien data näkymään. Näkymämalleissa muokataan mallien data sellaiseen muotoon, että se on helppo esittää näkymässä. Lisäksi näkymämallien tehtäviin kuulu päivittää malleja (Kaavio 3). Näkymämallit ovat C#-kielellä kirjoitettuja (Kuva 6). (MVVM)

```
namespace TAikaSeuranta.XClient.ViewModels
{
    5 references | Karoliina Jäppinen, 41 days ago | 1 author, 2 changes | 5 work items
    public class StaticInfoViewModel : BaseViewModel
    {
        //----------------------------------------------------------------
        1 reference | Karoliina Jäppinen, 41 days ago | 1 author, 1 change | 2 work items
        public StaticInfoViewModel(string text)
        {
            this.Text = text?.Trim();
        }
        //----------------------------------------------------------------
        private String _text;
        1 reference | Karoliina Jäppinen, 41 days ago | 1 author, 1 change | 2 work items
        public String Text
        {
            get { return _text; }
            set { SetProperty(ref _text, value); }
        }
        //----------------------------------------------------------------
    }
}
```

KUVA 6. Esimerkki yksinkertaisesta näkymämallista

3.4.3 Model (Malli)

Model eli malliluokat määrittelevät sovelluksen käyttämää dataa. Ne ovat C#-kielellä kirjoitettuja olioluokkia, eli datamalleja, jotka voivat sisältää myös sovelluksen logiikkaa (Kuva 7). Mallien luokkia käytetään usein palveluiden (service) tai hakemistojen (repository) kanssa datan käsittelyssä. (MVVM)

```
namespace TAikaSeuranta.XClient.Models
{
    3 references | Karoliina Jäppinen, 41 days ago | 1 author, 1 change | 2 work items
    public class PHIdentityInfo
    {
        1 reference | Karoliina Jäppinen, 41 days ago | 1 author, 1 change | 2 work items
        public Guid FaceId { get; set; }
        1 reference | Karoliina Jäppinen, 41 days ago | 1 author, 1 change | 2 work items
        public int? ID_Tyontekija { get; set; }
        1 reference | Karoliina Jäppinen, 41 days ago | 1 author, 1 change | 2 work items
        public String KokoNimi { get; set; }
    }
}
```

KUVA 7. Esimerkki yksinkertaisesta mallista

3.5 UWP

UWP eli Universal Windows Platform on yleinen sovellusalusta, joka mahdollistaa ohjelmien kehittämisen kaikille Windows 10 -pohjaisille laiteille. UWP:n avulla Xamarin.Formsilla tehdyn ohjelmiston saa toimimaan myös Windows-pohjaisilla laitteilla. (UWP)

3.6 Visual Studio 2019

Visual Studio on Microsoftin kehittämä integroitu kehitysympäristö (IDE), jolla voi kehittää ohjelmia esimerkiksi tietokoneille, mobiililaitteille sekä verkkoon. Visual Studio sisältää myös debug-ominaisuuden. Debug mahdollistaa koodin käyttäytymisen tarkastelun ohjelmaa ajaessa. Debugista on suuri hyöty mahdollisten ongelmien paikantamisessa sekä niiden selvittämisessä. Visual Studio 2019 on ohjelman uusin versio ja se toimi kehitysympäristönä työssäni. (VisualStudio)

3.7 Azure DevOps

Azure DevOps on Microsoftin kehittämä ohjelmistokehitystä tukeva ympäristö verkossa. AzureDevOps sisältää monia eri toimintoja, joista opinnäytetyössä käytettiin Azure Repos -versionhallintaa sekä Azure Boardsin Sprints-tauluja. Sprints-tauluun kirjattiin suoritettavat työtehtävät, jotka sitten pystyttiin linkittämään versionhallintaan. (DevOps)

Azure Repos on Git-periaatetta hyväksikäyttävä versionhallintaohjelmisto. Gitin toimintaperiaate on, että sovelluksen projektikansio, eli repository, on tallessa palvelimella, tässä tapauksessa Azuressa. Sieltä haetaan työkopio omalle koneelle, johon tehdään lisäyksiä sekä mahdollisia muutoksia sovellusta kehittäessä. Työkopio lähetetään takaisin palvelimelle, jossa se yhdistetään repositoryyn. Kaikki muutokset versioidaan, eli tarpeen mukaan voidaan palata takaisin edelliseen versioon, jos esimerkiksi nykyinen versio osoittauuu käyttökelvottomaksi. Gitin avulla on myös helpompaa kehittää sovellusta usemman henkilön toimesta.

3.8 NuGet

NuGet on pakentinhallinta, joka on suunniteltu .NET-ympäristölle. NuGetin avulla voidaan käyttää .NET-kirjastoja eli paketteja. Kirjastoja on myös mahdollista luoda ja jakaa muille käytettäväksi. Seuraavissa kappaleissa esitellään lyhyesti työssä käytetyt NuGet-paketit. (NuGet)

3.8.1 Autofac

Autofac on IoC-säiliö, joka on suunnattu .Net-ympäristölle. Sen avulla hallitaan luokkien riippuvuuksia toisiinsa. Autofac helpottaa sovellusten muokkaamista niiden kasvaessa ja muuttuessa monimutkaisemmiksi. (Autofac)

IoC-säiliö eli Inversion of Control -säiliö on kehys, jonka tarkoitus on toteuttaa automaattisesti riippuvuuksia luokkiin. Se hallitsee objektien luomisen ja hallinnan, joten niitä ei tarvitse tehdä manuaalisesti. (IoC)

3.8.2 Newtonsoft.Json

Newtonsoft.Json on JSON-kehys .NETille. Sen avulla pystytään muuntamaan dataa .Net-objektien ja JSON-tiedostojen välillä. JSON on kevyt datan muoto, jota käytetään datan tallennuksessa sekä siirrossa, esimerkiksi rajapinnat. JSONin avulla pystytään myös määrittämään yhdelle objektille useita ominaisuuksia. (Newtonsoft)

4 KÄYTTÖLIITTYMÄN TOTEUTUS

4.1 Valmistautuminen

Valmistautumisvaiheessa perehdyttiin työssä käytettäviin tekniikoihin. Xamarin.Formsin käyttöä opeteltiin lukemalla Microsoftin tekemiä dokumentaatiosivuja. Dokumentaatioiden lisäksi katsottiin tutorial-videoita, joissa näytettiin esimerkkisovellusten luomista Xamarin.Formsia käyttäen. Valmistautumisen aikana tutustuttiin myös Taika-ohjelmiston aiempaan koodiin ja sen toimintaan.

4.2 Suunnittelu

Työn tekeminen aloitettiin suunnittelulla. Suunnittelussa hyödynnettiin Taika-ohjelmiston vanhempaa versiota. Apua ja neuvoja saatiin myös kehitystiimin toiselta jäseneltä, joka oli alun perin kehittänyt Taikan. Suunnittelussa keskityttiin käyttöliittymän ulkoasuun ja navigointiin sekä projektin toteutukseen. Jo suunnitteluvaiheessa piti ottaa huomioon työn tilaajan toiveet siististä ja helposti ylläpidettävästä koodista. Työ rajattiin käyttöliittymän ja seurantatoiminnon toteuttamiseen.

4.2.1 Käyttöliittymä

Käyttöliittymää suunnitellessa sen värimaailma ja teema määräytyivät suoraan ohjelman edellisen version mukaiseksi. Teemaa suunnitellessa täytyi ottaa huomioon ohjelman ylläpidettävyys, sillä teemaa täytyy pystyä tarvittaessa vaihtamaan. Teema päätettiin tehdä resurssimenetelmällä, eli yhteen tiedostoon kasattiin kaikki värien, fonttien ja käyttölittymän palasten ulkoasun määritelmät. Näkymissä käytettiin kyseisessä tiedostossa määriteltyjä muuttujia. Tällä tavalla teemaa vaihtaessa tai muokattaessa tarvitsee muokata vain yhtä tiedostoa.

Xamarin.Forms-projektissa on mahdollista rakentaa navigointi monella eri tavalla. Työssä päädyttiin käyttämään MasterDetailPage-navigointia. MasterDetailPagessa on kaksi toisiinsa liityvää sivua, master- ja detail-sivu. Master-sivulla on tässä tapauksessa sivusta aukeava valikko, jossa on lista käyttöliittymän sivuista (Kuva 8). Detail-sivu puolestaan näyttää valitun sivun. (MasterDetailPage)

```
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
                  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                  xmlns:d="http://xamarin.com/schemas/2014/forms/design"
                  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
                       xmlns:views="clr-namespace:TAikaSeuranta.XClient.Views"
                       mc:Ignorable="d"
                       xmlns:models="clr-namespace:TAikaSeuranta.XClient.Models"
                       models:ViewModelLocator.AutoWireViewModel="True"
                       MasterBehavior="Popover"
                  x:Class="TAikaSeuranta.XClient.Views.MainView">
    <MasterDetailPage.Master>
        <views:MenuView></views:MenuView>
    </MasterDetailPage.Master>
    <MasterDetailPage.Detail>
        <ContentPage></ContentPage>
    </MasterDetailPage.Detail>
</MasterDetailPage>
```

KUVA 8. MasterDetaliPagen XAML-koodi, josta näkyy valittu Master-sivu, MenuView.

4.2.2 Projektin toteutus

Projektin toteutuksen suunnittelussa eriteltiin käyttöliittymän kehitys kokonaisuuksiin. Kokonaisuudet kirjattiin työtehtävinä Azure DevOpsin Sprint-tauluun, josta ne on mahdollista linkittää versionhallinnassa käytettäviin pull requesteihin eli vetopyyntöihin. Kun sopiva kokonaisuus saadaan tehtyä, tehdään siitä vetopyyntö, johon liitetään siinä tehdyt työtehtävät. Kehitystiimin vastaava henkilö käy sitten muutokset läpi ja hyväksyy tai hylkää pyynnön. Pyyntöön voi myös lisätä kommentteja, jos jotain korjattavaa löytyy.

4.3 Kehittäminen

Käyttöliittymän kehittäminen tapahtui pääosin kotona itsenäisesti. Koska toimisto sijaitsi toisella paikkakunnalla, siellä pyrittiin käymään vähintään kerran viikossa. Toimistolla käydessä tarkasteltiin kehityksen eteneminen sekä mahdolliset ongelmakohdat. Ongelmia pystyttiin ratkomaan myös etänä Microsoft Teams -sovelluksen avulla. Teams tarjoaa mahdollisuuden näytönjakamiseen puhelun aikana. Näytönjakamisesta oli suuri apu ongelmien ratkaisussa, sillä silloin oli mahdollista esitellä ongelmakohdat reaaliaikaisesti sekä ratkoa niitä yhdessä.

4.3.1 Käyttöliittymän pohja ja navigointi

Aluksi käyttöliittymään toteutettiin navigoinnin pohja. Navigointia varten tehtiin MasterDetailPagen vaatima valikkorakenne sekä Navigaatiopalvelu (service). Navigaatiopalvelu hoitaa navigoinnin ohjaamisen. Lisäksi tehtiin valmiiksi pohjat tarvittaville näkymille ja näkymämalleille.

Ensimmäinen lisätty toiminto oli sisäänkirjautuminen. Sisäänkirjautuminen toimii lisenssillä. Tässä vaiheessa lisenssinä toimi keksitty numerosarja, jolla testattiin sisäänkirjautumisen ja aloitussivulle navigoimisen toimivuus.

Seuraavassa vaiheessa siirrettiin kaikki tarvittavat kooditiedostot Taikan vanhasta koodista. Nämä tiedostot sisälsivät suuren osan Taikan logiikasta. Tiedostoissa oli paljon muokattavaa, sillä Xamarin.Forms käyttää erilaisia funktioita ja kirjastoja kuin Taikan vanha versio.

4.3.2 Teema ja ulkoasu

Näkymien ulkoasut määriteltiin XAML-kieltä käyttäen. Vaikka Taikan vanha käyttöliittymä oli myös tehty XAML:lla, sitä ei voitu suoraan siirtää uuteen. Xamarin.Forms käyttää erilaisia UI-kontrolleja kuin WPF. Joitakin kontrolleja ei ollut Xamarin.Formsissa lainkaan. Myös kontrollien ominaisuudet olivat erilaisia. Ulkoasu tehtiin siis uudestaan vanhaa apuna käyttäen.

Teema rakennettiin myös XAML-kielellä. Teema tehtiin App.xaml tiedoston Application.Resources-osioon. Sinne määriteltiin teeman käyttämät värit sekä kontrolleille halutut ominaisuudet (Kuva 9). Näkymät pystyivät käyttämään määriteltyjä resursseja niiden Key-ominaisuuden avulla (Kuva 10).

```
Application                                                    Application
1   <?xml version="1.0" encoding="utf-8" ?>
2   <Application xmlns="http://xamarin.com/schemas/2014/forms"
3                xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4                xmlns:d="http://xamarin.com/schemas/2014/forms/design"
5                xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6                mc:Ignorable="d"
7                x:Class="TAikaSeuranta.XClient.App">
8       <Application.Resources>
9           <!--COLORS-->
10
11          <Color x:Key="basicColor1">#f4f4f4</Color>
12          <Color x:Key="basicColor2">#8597cb</Color>
13          <Color x:Key="basicColor3">#f4f4f4</Color>
14          <Color x:Key="basicColor3lighter">White</Color>
15          <Color x:Key="basicColor4">#FF234C7C</Color>
16          <Color x:Key="basicColor5">#1c2a58</Color>
17          <Color x:Key="basicTextColor1">Black</Color>
18          <Color x:Key="basicTextColor2">White</Color>
19          <Color x:Key="basicTextDisabledColor1">#FF74777B</Color>
20          <Color x:Key="nodeExecutingColor1">#FFC44B26</Color>
21          <Color x:Key="basicSelectedColor1">#009900</Color>
22          <Color x:Key="basicColorWhite">White</Color>
23
24          <!--CONTROLS-->
25
26          <Style TargetType="{x:Type StackLayout}">
27              <Setter Property="Spacing" Value="0"></Setter>
28          </Style>
29
30          <Style TargetType="{x:Type Grid}">
31              <Setter Property="BackgroundColor" Value="{StaticResource basicColor1}"></Setter>
32          </Style>
33
```

KUVA 9. App.xaml tiedoston alku, värien ja muutaman kontrollin määritys

```
<Frame Margin="5" BorderColor="{StaticResource basicColor5}">
    <Entry Text="{Binding Tiedot, FallbackValue=[TIEDOT]}"
           FontSize="22"
           HorizontalTextAlignment="Start"
           VerticalTextAlignment="Start">
    </Entry>
</Frame>
```

KUVA 10. Resurssien käyttö näkymässä

Näkymien ulkoasun kehityksessä oli apuna myös Visual Studion ominaisuus, XAML Previewer. Previewerin avulla pystyttiin kehitysvaiheessa näkemään, mille kirjoitettu XAML-koodi näyttäisi Android-laitteella.

4.3.3 Testaus ja viimeistely

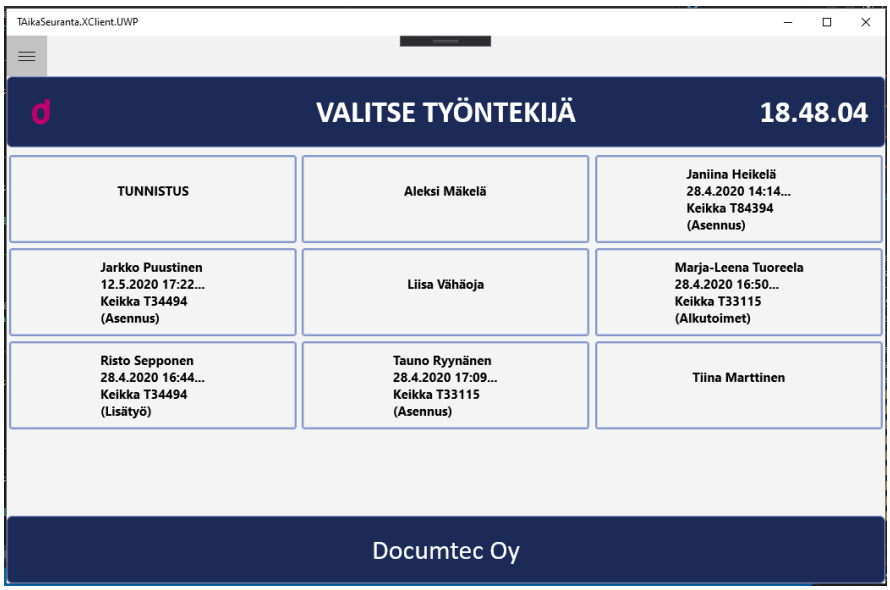
Käyttöliittymän toiminnan testausta varten sisäänkirjautuminen muutettiin käyttämään oikeaa lisenssiä. Lisenssinä toimi testaustarkoitukseen tehty lisenssi, jonka avulla saatiin testidataa haettua palvelimelta. Kuten aiemmin jo mainittiin, Taikan käyttöliittymä tarvitsee palvelimen logiikkaa toimiakseen.

Testauksessa käytettiin Visual Studion debug-ominaisuutta, jonka avulla pystyttiin ohjelmaa ajamaan oman koneen sekä Android-emulaattorin kautta. Omalla koneella lokaalisti ajamalla pystyttiin testaamaan, miltä ohjelma näyttäisi muillakin Windows-laitteilla. Android-emulaattori tarkoittaa virtuaalista mobiililaitetta. Emulaatoreita oli saatavilla monia eri vaihtoehtoja, esimerkiksi eri kokoisilla näytöillä sekä eri Android-versioilla. Emulaattorin avulla pystytiin testaamaan käyttöliittymän toimintaa mobiililaitteissa.

Lopuksi käyttöliittymän ulkoasu vaati vielä hienosäätöä. Tämä johtui siitä, että ennen testidatan saamista, käyttöliittymän testaaminen ei ollut mahdollista. Käyttöliittymän palasten kokoja piti muokata oikean kokoisiksi, kaikkien tekstien mahtuminen tekstikenttiin täytyi varmistaa sekä värit ja fontit tarkistettiin.

4.4 Lopputulos

Lopputuloksena saatiin uudistettu käyttöliittymä Taika-ohjelmistolle (Kuva 11). Uusi käyttöliittymä toimii myös toivotusti Android-pohjaisilla mobiililaitteilla (Kuva 12). Työaikaseurantatoiminto toimii kuten pitääkin. Työntekijä pystyy kirjautumaan sisään ja ulos töistä. Lisäksi työtehtävän pystyy valitsemaan sisäänkirjautuessa tai tarvittaessa vaihtaa kesken päivän.



KUVA 11. Taikan perusnäkymä uudella käyttöliittymällä

KUVA 12. Taikan perusnäkymä uudella käyttöliittymällä mobiililaitteessa

Uuden käyttöliittymän ulkoasu mukailee paljon Taikan vanhaa versiota. Uutena ominaisuutena on valikkonäkymä, joka aukeaa käyttöliittymän vasemmassa yläkulmassa sijaitsevasta napista (Kuva 13). Valikon avulla pystytään kätevästi navigoimaan Seuranta-näkymän sekä tulevaisuudessa lisättävän Omat tiedot-näkymän välillä. Valikkoon pystyy myös tarvittaessa lisämään helposti valittavien näkymien joukkoon uusia.

KUVA 13. Taikan uusi valikkonäkymä

Lopputuloksen saavuttamiseksi täytyi hyödyntää monia taitoja. Eniten hyötyä oli itsenäisen työskentelyntaidosta, sillä opinnäytetyön eteneminen oli täysin itsestä kiinni. Aikataulun suunnittelu oli myös keskeisessä osassa, opinnäytetyön raportointi oli tärkeää aloittaa jo hyvissä ajoin kehitystyön rinnalle, ettei loppuvaiheessa tulisi turhaa kiirettä sen kanssa.

5 POHDINTA JA YHTEENVETO

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa Taika-ohjelmistolle uusi käyttöliittymä. Uudelle käyttöliittymälle oli todellinen tarve, koska edellinen versio ei tukenut mobiililaitteita. Lopputuloksena syntyi Xamarin.Formsin avulla kehitetty käyttöliittymä, joka vastasi vaadittuja tavoitteita.

Opinnäytetyötä suunnitellessa työ rajattiin käyttöliittymän ja seurantatoiminnon toimivuuteen. Työajanseuranta on Taikan tärkein toiminto. Käyttöliittymää tullaan vielä jatkokehittämään siten, että siihen lisätään muut Taikan toiminnot.

Työn tekeminen eteni suunnitelmien mukaisesti ja varattu aika riitti juuri sopivasti. Haasteita työn etenemiseen aiheutti Korona-virus, joka vaivasi koko maailmaa kevään 2020 aikana. Viruksen takia jouduttiin vaihtamaan työnteko kokonaan etätyöskentelyksi. Ongelmatilanteiden ratkaisu yhdessä muiden kanssa jouduttiin hoitamaan etänä, mikä ei ollut yhtä tehokasta kuin toimistolla samassa tilassa työskennellessä.

Työn alkuvaiheessa haasteita aiheutti Xamarin.Forms, sillä se oli minulle täysin uusi kehitysmenetelmä. Sen toimintaperiaate kuitenkin opittiin työtä tehdessä, minkä jälkeen kehitystyö alkoi sujua paremmin. Hankaluuksia tuotti myös Taikan logiikan ymmärtäminen ja siirtäminen uuteen ympäristöön. Tässä apuna toimi kehitystiimin toinen jäsen, joka selitti ja auttoi minua tarvittaessa.

Opinnäytetyötä tehdessä jouduin soveltamaan paljon jo opittuja taitoja sekä oppia uusia. Opin paljon uutta ohjelmoinnista sekä työssä käytetyistä tekniikoista. Pääsin perehtymään paremmin .NET:n maailmaan sekä opin uuden tavan kehittää mobiilisovelluksia. Lisäksi sain lisää kokemusta projektinhallinnasta ja itsenäisestä työskentelystä.

LÄHTEET

.NET. What is .NET? [Online] [Viitattu: 14. 05 2020.] Saatavissa: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet.

Autofac. Autofac. [Online] [Viitattu: 25. 04 2020.] Saatavissa: https://www.nuget.org/packages/Autofac/.

C#. What Is C#. [Online] [Viitattu: 04. 04 2020.] Saatavissa: https://www.c-sharpcorner.com/article/what-is-c-sharp/.

CognitiveServices. What is Azure Cognitive Services? [Online] [Viitattu: 22. 04 2020.] Saatavissa: https://azure.microsoft.com/en-us/services/cognitive-services/#features.

DevOps. Azure DevOps. [Online] [Viitattu: 04. 04 2020.] Saatavissa: https://azure.microsoft.com/en-us/services/devops/.

Documtec. Documtec. [Online] [Viitattu: 11. 04 2020.] Saatavissa: https://www.documtec.fi/.

IoC. IoC Container. [Online] [Viitattu: 25. 04 2020.] Saatavissa: https://www.tutorialsteacher.com/ioc/ioc-container.

MasterDetailPage. Xamarin.Forms Master-Detail Page. [Online] [Viitattu: 17. 04 2020.] Saatavissa: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/navigation/master-detail-page.

MVVM. The Model-View-ViewModel Pattern. [Online] [Viitattu: 20. 04 2020.] Saatavissa: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

Newtonsoft. JSON.NET Documentation, Introduction. [Online] [Viitattu: 25. 04 2020.] Saatavissa: https://www.newtonsoft.com/json/help/html/Introduction.htm.

NuGet. What is NuGet? [Online] [Viitattu: 24. 04 2020.] Saatavissa: https://www.nuget.org/.

UWP, in Xamarin. Xamarin and the Universal Windows Platform. [Online] [Viitattu: 08. 04 2020.] Saatavissa: https://docs.microsoft.com/en-us/archive/msdn-magazine/2016/connect/xamarin-xamarin-and-the-universal-windows-platform.

VisualStudio. Visual Studio 2019. [Online] [Viitattu: 04. 04 2020.] Saatavissa: https://visualstudio.microsoft.com/vs/.

WPF. Get Started (WPF). [Online] [Viitattu: 22. 04 2020.] Saatavissa: https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/.

Xamarin. What is Xamarin? [Online] [Viitattu: 08. 04 2020.] Saatavissa: https://docs.microsoft.com/fi-fi/xamarin/get-started/what-is-xamarin.

Xamarin.Forms. What is Xamarin.Forms? [Online] [Viitattu: 08. 04 2020.] Saatavissa: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms.

XAML. Xamarin.Forms XAML Basics. [Online] [Viitattu: 04. 04 2020.] Saatavissa: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xaml-basics/.